SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**SEMINAR**

# HRCM algoritam za kompresiju genoma

*Hana Jurić Fot i Jan Novosel*
Voditelj: *Mirjana Domazet-Lošo*

Zagreb, svibanj 2023.

# SADRŽAJ

# 1. Uvod

Projekt određivanja ljudskog genoma, koji je započeo 1990., a završio 2003. godine, potaknuo je snažan razvoj metoda za sekvenciranje (Domazet-Lošo i Šikić, 2014). Razvojem novih metoda drastično je počela padati cijena sekvenciranja, što je dovelo do sve većeg broja genomskih podataka. Sve te podatke potrebno je negdje pohraniti. Najčešće korišten format za pohranu genomskih podataka je FASTA tekstualni format, u kojemu su sljedovi nukleinskih kiselina prikazani tako da je svaki nukleoid kodiran jednim slovom. Svaki zapis u FASTA datoteci sastoji se od linije zaglavlja i linija slijeda, pri čemu redak zaglavlja započinje znakon '>' i sadržava identifikator slijeda (Domazet-Lošo i Šikić, 2014). Budući da je ovaj format zapisa vrlo raširen, a broj genomskih podataka izrazito brzo raste, javila se potreba za kompresijom FASTA formata.

Za kompresiju se mogu koristiti algoritmi komprsije opće namjere poput PPMD ili DEFLATE algoritama, no oni ne uzimaju u obzir karakteristike genoma te ne mogu postići velik omjer kompresije. Zato su se pojavili algoritmi za kompresiju genoma posebne namjere (engl. *special purpose genome compression algorithms*). Ovi algoritmi dijele se u dvije kategorije: bez i s referencom. Algoritmi bez reference dalje se dijele na naivne algoritme za kodiranje bitova, algoritme temeljene na rječniku i statističke algoritme.

Naš zadatak na ovom projektu bio je proučiti algoritam HRCM (engl. *hybrid referential compression method*) (Yao et al., 2019) i implementirati ga. Ovaj algoritam spada u algoritme za kompresiju genoma s referencom, a nakon takve kompresije se podaci još jednom kompresiraju koristeći algoritam za kompresiju opće namjere.

# 2. HRCM algoritam

HRCM (engl. *hybrid referential compression method*) učinkovit je algoritam za kompresiju genomskih podataka u FASTA formatu, a sastoji se od sljedeća tri koraka:

1. Izvlačenje informacija koje se ne komprimiraju
2. Kompresija prve razine
3. Kompresija druge razine

## 2.1. Izvlačenje informacija

U ovom je koraku potrebno iz ciljnih nizova izvući informacije o pozicijama malih slova, znakova `N` i ostalih znakova zato što se te informacije ne komprimiraju. Ove informacije pohranjuju se zajedno s komprimiranom datotekom i koriste u zadnjem koraku dekompresije za rekonstrukciju početnih nizova. Svi se znakovi osim `A`, `C`, `G` i `T` brišu, a mala se slova pretvaraju u velika.

## 2.2. Kompresija prve razine

Prva se razina svodi na reprezentaciju niza koji komprimiramo (ciljnog niza) putem podnizova koji su mu zajednički s referentnim nizom. Što su ciljni i referentni niz sličniji, kompresija će biti bolja. Rezultat kompresije prve razine su trojke $(pozicija$ ,

| | |
|---|---|
| Niz: | `ganCTGATaagtCXxagGACnNNAG` |
| Mala slova: | $(0, 3)$, $(8, 4)$, $(14, 3)$, $(20, 1)$ |
| Znakovi `N`: | $(2, 1)$, $(20, 1)$ |
| Ostali znakovi: | $(13, X)$, $(14, X)$ |

**Tablica 2.1:** Izvlačenje informacija

| | |
|---|---|
| Referentni niz: | GATCTGATAAGTCCCAGGACTTCAG |
| Ciljni niz 1: | GATCTGATAGGTCCCAGGACTTCAG |
| Ciljni niz 2: | GATCTGATAAGTCCCATGACTTCAG |
| Ciljni niz 3: | GATATGAAAAGTCACAGGAAAACAG |
| Ciljni niz 4: | GATCTGATGATTACAAGGACTTCGG |
| Ciljni niz 5: | GATCTGATAAGTCCCAGGACCCCCC |

**Tablica 2.2:** Prva razina - ulaz

| | |
|---|---|
| Trojke 1: | $(0, 9, G),\ (10, 31, )$ |
| Trojke 2: | $(0, 9, G),\ (10, 6, T),\ (17, 9, )$ |
| Trojke 3: | $(5, 4, ),\ (4, 3, A),\ (8, 5, A),\ (14, 5, ),\ (8, 2, A),\ (22, 18, )$ |
| Trojke 4: | $(0, 8, ),\ (5, 3, ),\ (7, 2, ),\ (22, 2, ),\ (15, 8, ),\ (16, 2, )$ |
| Trojke 5: | $(0, 8, ),\ (5, 3, ),\ (7, 2, ),\ (22, 2, ),\ (5, 5, ),\ (12, 3, ),\ (13, 2, )$ |

**Tablica 2.3:** Prva razina - izlaz

$duljina$ , $nepodudaranje$). Svaka trojka predstavlja podudarni podniz te govori gdje se taj podniz nalazi u referentnom nizu, koliko je dug i kako glasi nepodudaranje nakon njega. Izlaz je $n$ nizova trojki gdje je $n$ broj ciljnih nizova. Kako bi pronalazak podudaranja bio brži, temeljem referentnog niza stvaramo hash tablicu `H[]` i polje povezivanja `L[]` metodom hashiranja $k$-mera.

`value_i` : hash vrijednost i-tog k-mera referentnog niza

```
L[i] = H[value_i]
H[value_i] = i
```

Podudaranja (trojke) upisujemo redom od najduljeg prema najkraćem, a potom ih sortiramo prema početnim indeksima u ciljnom polju. Svi znakovi koji nisu pokriveni podudaranjima upisujemo kao nepodudaranja u odgovarajuće trojke.

U tablici 2.2 prikazano je pet sekvenci koje želimo kompresirati te njihova referentna sekvenca, a u tablici 2.3 prikazane su odgovarajuće trojke nakon prve razine kompresije. Ako kompresiramo samo jednu sekvencu, onda smo gotovi, odnosno prelazimo na kompresiju opće razine, no ako imamo više sekvenci, trebamo raditi i drugu razinu kompresije.

| | |
|---|---|
| Trojke 1: | $(0, 9, G)$, $(10, 31, )$ |
| Trojke 2: | $(0, 0, 1)$, $(10, 6, T)$, $(17, 9, )$ |
| Trojke 3: | $(5, 4, )$, $(4, 3, A)$, $(8, 5, A)$, $(14, 5, )$, $(8, 2, A)$, $(22, 18, )$ |
| Trojke 4: | $(0, 8, )$, $(5, 3, )$, $(7, 2, )$, $(22, 2, )$, $(15, 8, )$, $(16, 2, )$ |
| Trojke 5: | $(3, 0, 4)$, $(12, 3, )$, $(13, 2, )$ |

**Tablica 2.4:** Druga razina - izlaz

## 2.3. Kompresija druge razine

Druga se razina svodi na reprezentaciju svakog dobivenog niza trojki putem ostalih nizova trojki proizašlih iz prve razine. Pretpostavlja se da i među njima ima sličnosti i cilj je temeljem njih provesti još jednu razinu kompresije. Ponovno za svaki ulazni niz generiramo izlazni niz, no sada referencom za $i$-ti niz trojki smatramo prvih $(i-1)$ nizova. Potrebno je stvoriti hash tablicu za svaki od $n$ nizova `H[][]`.

`value_i` : hash vrijednost i-te trojke m-tog niza trojki

```
H[m][value_i] = i
```

Sada imamo dvije vrste trojki: podudaranje i nepodudaranje. Trojka podudaranja je oblika $(m\ ,\ pozicija\ ,\ duljina)$, gdje je $m$ indeks referentnog niza trojki (odnosno indeks sekvence iz koje je ta trojka došla), $pozicija$ je pozicija podudaranja u referentnom nizu, a $duljina$ je broj uzastopnih podudarnih trojki. Trojka nepodudaranja istog je oblika kao i trojke iz prve razine zato što trojke koje postoje jedino u nizu koji komprimiramo jednostavno prepisujemo. Podudaranja $i$-tog niza, dakle, tražimo u prvih $(i-1)$ referentnih nizova, a zapisujemo ono najdulje.

Tablica 2.4 prikazuje izlaz nakon druge razine kompresije.

## 2.4. Implementacija i rezultati

Naša implementacija razlikuje se od originalne u nekoliko stvari. Naša implementacija ne uključuje kompresiju informacija o pozicijama malih slova. Nadalje, originalna implementacija koristi drugačiju hash funkciju i drugačiji postupak pronalaska podudaranja što omogućuje manje memorijsko zauzeće i manje vrijeme kompresije.

Implementaciju algoritma ispitali smo na genomima reda veličine $10^3$, $10^4$ i $10^5$ znakova. Genomi reda veličine $10^4$ znakova su genomi HIV-a, a ostali su podaci podnizovi genoma E. coli. Testiranje je provedeno na računalu opremljenom 12th Gen

| Vrsta | # znakova | # ciljnih nizova | Vrijeme kompresije (s) | Zauzeće memorije (MB) | Kompresija (%) |
|---|---|---|---|---|---|
| E. coli | 3969 | 3 | 1 | 957.4 | 87.79 |
| E. coli | 3969 | 4 | 1 | 957.4 | 87.84 |
| HIV | 9843 | 2 | 2 | 957.4 | 76.78 |
| HIV | 9843 | 3 | 2 | 957.4 | 75.10 |
| HIV | 9843 | 4 | 3 | 957.4 | 74.83 |
| E. coli | 404999 | 4 | 5601 | 957.4 | 80.67 |

**Tablica 2.5:** Performanse

Intel® Core™ i7-1255U × 12 procesorom i 16 GB radne memorije. Kao što se vidi u tablici 2.5, zauzeće memorije ne ovisi o broju znakova genoma niti o broju ciljnih nizova. Memorijski dominantna struktura je polje 32-bitnih brojeva `H[]` koje pri kompresiji svakog od ciljnih nizova zauzima $4 \times 5^k$ bajtova, što za $k = 12$ koji koristimo u našoj implementaciji otprilike odgovara maksimalnom memorijskom zauzeću programa u testnim primjerima. Vremena dekompresije su $< 1$ sekunde za sve primjere.

# 3. Zaključak

Algoritam kompresije specijaliziran za kompresiju genoma puno je učinkovitiji od algoritama kompresije za opću namjenu. Primjerice, HRCM kompresija jednog genoma reda veličine $10^5$ znakova (oko 4 kB memorije) kompresijom prve razine dala je datoteku veličine oko 0.25 B, dok je kompresija istog genoma samo DEFLATE algoritmom dala datoteku veliku 1.6 kB. Na ovom primjeru možemo vidjeti da je HRCM kompresija zaista bolja od kompresija opće namjene.

# 4. Literatura

Mirjana Domazet-Lošo i Mile Šikić. *Bioinformatika - skripta*. 2014.

Haichang Yao, Yimu Jo, Kui Li, Shangdong Liu, Jing He, i Ruchuan Wang. Hrcm: An efficient hybrid referential compression method for genomic big data. 2019.